IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application No.: 10/726,902	§		
Filed: December 3, 2003	§	Examiner:	Fennema, Robert E.
Inventors:	§	Group/Art Unit:	2183
Mitchell Alsup, et al.	§	Atty. Dkt. No:	5500-88700
Title: Transitioning From Instruction	§		
Cache to Trace Cache on	§		
Label Boundaries			

PRE-APPEAL BRIEF REQUEST FOR REVIEW

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Applicants request review of the final rejection in the above-identified application. Claims 1-35 are pending in the application. Reconsideration of the present case is earnestly requested in light of the following remarks.

The Examiner rejected claims 1, 2, 11-15 and 24-26 under 35 U.S.C. § 102(b) as being anticipated by Rotenberg, et al. (hereinafter "Rotenberg"). The Examiner rejected claims 3 and 16 under 35 U.S.C. § 103(a) as being unpatentable over Rotenberg in view of Patterson, et al. (hereinafter "Patterson"), claims 4, 10, 17 and 23 as being unpatentable over Rotenberg in view of Braught, and further in view of Xia, claims 5-8 and 18-21 as being unpatentable over Rotenberg, Xia, Braught and further in view of Lange, et al. (U.S. Patent 3,896,419) (hereinafter "Lange"), claims 9 and 22 as being unpatentable over Rotenberg, Xia and Braught in view of Akkary, et al. (U.S. Patent 6,247,121) (hereinafter "Akkary"), claims 27-31 and 35 as being unpatentable over Rotenberg, Xia, Braught and Lange in view of Akkary, and claims 32 and 34 as being unpatentable over Rotenberg in view of Xia. Applicants note the following clear errors in the Examiner's rejection.

Independent claim 1:

1. The cited art clearly fails to disclose *wherein the prefetch unit is configured to check the trace cache for a match for the predicted target address in response to the branch prediction unit outputting a predicted target address; and wherein the prefetch unit is configured to not check the trace cache for a match until the branch prediction unit outputs the predicted target address.*

In the system of Rotenberg, "The trace cache is accessed in parallel with the instruction cache and BTB using the current fetch address..." (emphasis added.) **It is clear in this and other passages previously cited by Applicants that in Rotenberg every fetch address is compared to the entries in the trace cache, regardless of the operation of the branch prediction unit.** Rotenberg teaches searching for a hit in the trace cache on every cycle, and then fetching from the instruction cache if there is not a hit in the trace cache. By contrast, Applicants' claim 1 recites *wherein the prefetch unit is configured to not check the trace cache for a match **until** the branch prediction unit outputs the predicted target address*. **In other words, the trace cache is not checked for a match on every cycle, but only on cycles for which the branch prediction unit outputs a predicted target address.** The Examiner previously submitted, "It is inherent that you cannot check for a value if you do not know what the value is. The trace cache as disclosed by Rotenberg cannot search for the predicted target address in the cache if it does not know what the predicted target address is, and in fact, no cache or any other type of hardware can find something when it doesn't know what it is looking for, thus, the value must have been output from the branch prediction unit prior to the checking." Applicants assert that the Examiner's interpretation of the operation of Rotenberg is demonstrably incorrect since Rotenberg explicitly checks for a match on every cycle, regardless of the operation of the branch prediction unit and whether it outputs a predicted target address.

In the Final Action mailed July 18, 2007, the Examiner submitted, "When looking at the entire claim in context, it can be seen that the claim is referring to searching the cache for a match for the predicted target address, and that it cannot search the cache for the predicted target address until the predicted target address is generated" (emphasis Examiner's). **The Examiner has misread the claim.** The referenced limitation of the claim does not recite "not check the trace cache for the match" (i.e., referring to the match for the predicted target address of the previous limitation), but instead recites "not check the trace cache for a match" (i.e., any match between the current address and the trace cache). **The plain language of the claim does not support the Examiner's interpretation. As explicitly recited in claim 1,** multiple instructions are fetched from the instruction cache, **without checking for a match (hit) in the trace cache**, until the branch prediction unit outputs a predicted branch address (i.e., until a branch instruction is encountered for which the branch prediction unit outputs a predicted target address). Only then (**in response to this output**) is the trace cache checked for a match. Applicants also assert that one of ordinary skill in the art having benefit of Applicants' disclosure could not agree with the Examiner's interpretation of the referenced limitation. The Examiner's repeated assertions that it is impossible to search for something that has not yet been generated are irrelevant to Applicants' argument and to what is actually recited in the claim. The Examiner's interpretation is also completely inconsistent with the specification. Also, in remarks in the Final Action regarding claim 32, **the Examiner admits that Rotenberg does not teach not searching the trace cache until a branch target address is generated**, instead teaching that the trace cache is searched on every instruction. Therefore, Rotenberg clearly does not anticipate the above-referenced limitations of Applicants' claim. Claim 14 includes limitations similar to claim 1, and so the arguments presented above apply with equal force to this claim, as well.

Independent claim 27:

1. The cited art clearly fails to teach or suggest *receiving a retired instruction.*

The Examiner admits that Rotenberg does not teach that instructions need to be retired before the trace can be generated and relies on Akkary to teach this limitation. The Examiner submits that Akkary teaches that instructions are not put into the trace buffers until they have been retired, to ensure that they executed correctly (column 3, lines 40-44). **This passage actually says just the opposite**, that is, that instructions placed in the trace buffer stay in the trace buffer until they are retired.

2. The cited references fail to teach or suggest *in response to determining that a previous trace under construction duplicates a trace in a trace cache and that the received instruction corresponds to a branch label, beginning construction of a new trace.*

The Examiner admitted that Rotenberg fails to teach this limitation or to discuss the issue of duplication. The Examiner submits that Rotenberg discusses the disadvantages which occur when a trace cache miss occurs while servicing a previous trace cache miss, and teaches the disadvantages of a useless trace displacing a useful trace. The Examiner then submits, "Therefore, Rotenberg teaches a system in which allows tracing of potential duplicate traces, and also a system which requires action when a miss occurs while servicing a miss." Applicants assert that the Examiner is contradicting himself and that he has cited nothing in Rotenberg that teaches tracing of potential duplicate traces. In Rotenberg, the fill-line buffer logic services trace cache misses (i.e., the combination of a matching target address and matching branch flags is not found in the trace cache). There is no reason to believe there would ever be a duplicate trace in the system of Rotenberg, and no reasons or opportunity to avoid such duplication. The Examiner suggests that there is a "potential for duplicate traces to exist with path associativity in Rotenberg's alternative embodiments." **This is completely unsupported in Rotenberg and does nothing to overcome the deficiencies of the cited references in teaching the above-referenced limitations. Similarly, the Examiner's contention that "Rotenberg indicates in his judicial trace selection that storing a duplicate trace would be at best useless, and at worst displace a useful trace that may be used" is also completely unsupported.** The solution discussed in this section is completely different from the specific limitations recited in claim 27. The Examiner submits that this section provides motivation to handle cases involving duplicate traces, and that Lange provides such a method, by simply discarding the duplicate value. **Again, the Examiner's remarks do not address the limitations recited in claim 27, which are directed to specific conditions to be met before beginning construction of a new trace, and which the cited references do not teach.**

3. The Examiner has failed to provide a proper reason to combine the references.

The Examiner argues that Akkary suggests the limitation of using a retired instruction, and he includes remarks regarding "correctness" of a trace, a goal of Rotenberg's trace cache to exploit code reuse, and that branches tend to be biased in one direction. However, Rotenberg explicitly states that the trace line is filled as instructions are fetched from the instruction cache, **clearly teaching away from filling the trace line with retired instructions**. The fact that the system of Rotenberg could solve this problem in a manner different from the only example described does not suggest the specific solution recited in the claims. **Applicants also assert that combining the references does not teach the claimed invention,** as neither reference teaches a solution that involves receiving a retired instruction, and in response to determining that a previous trace under construction duplicates a trace in a trace cache and that the received instruction corresponds to a branch label, **beginning construction of a new trace.** The Examiner previously cited Lange's description of the operation of a data cache during fetching from main memory. Applicants assert that this has absolutely nothing to do with the specific limitations of claim 27. Checking Rotenberg's trace cache for a duplicate trace before collecting a new trace is also contradictory to the Examiner's remarks above regarding the advantages of including duplicate traces. **The Examiner first argues that Rotenberg teaches duplicate traces may exist, and then argues that the combined references teach that the system of Rotenberg should not allow construction of duplicate traces. These arguments cannot coexist if the references are to teach the limitations of claim 27. If no duplicate traces can be constructed, there would be no reason to determine if a trace previously under construction was duplicating a trace already in the trace cache.** The Examiner's reasons for combining the references are clearly not supported by the cited art, and the references, when combined, do not teach the limitations recited in claim 27. For at least the reasons above, the rejection of claim 27 is unsupported by the cited art and removal thereof is respectfully requested. Claim 35 includes limitations similar to claim 27, and so the arguments presented above apply with equal force to this claim, as well.

Independent Claim 32:

1. The cited art clearly fails to disclose *a method, comprising: fetching instructions from an instruction cache; continuing to fetch instructions from the instruction cache without searching a*

trace cache until a branch target address is generated; in response to a branch target address being generated, searching a trace cache for an entry corresponding to the branch target address.

The Examiner submits that Rotenberg teaches all of the limitations of this claim except, "without searching a trace cache" and relies on Xia to teach this limitation. However, Rotenberg does not teach searching a trace cache in response to a branch target address being generated. **In fact, the Examiner admits that Rotenberg teaches that the trace cache is searched on every instruction.** Applicants note Xia's trace table and instruction cache are also checked in parallel on every instruction. **The Examiner's contention that it would be obvious not to search the trace cache for a non-branch instruction is completely unsupported, as his own references check the trace cache on every instruction. His assertion that such a technique would contribute to power saving or critical path reduction is similarly unsupported in the cited art.** The only advantage noted for starting traces on branches (i.e., to save memory space) does not require a change to the method for searching the trace cache. There is nothing in Xia that describes how, or more importantly when, the trace cache is checked for the start of a trace line, or when the system begins a search of the trace cache for any reason. The Examiner's statement, "The advantages that Examiner indicated… are obvious advantages one of ordinary skill in the art would recognize, searching a cache takes time and energy, and if there is no possible way that searching a cache can benefit the user, there is absolutely no reason to do so" contradict the teachings of his own references, which search the cache for a hit (although not necessarily a trace start) on every cycle. **This is in direct conflict with the limitations of claim 32.** For at least the reasons above, the rejection of claim 32 is unsupported by the cited art, and removal thereof is respectfully requested.

In light of the foregoing remarks, Applicants submit the application is in condition for allowance, and notice to that effect is respectfully requested. If any extension of time (under 37 C.F.R. § 1.136) is necessary to prevent the above referenced application from becoming abandoned, Applicants hereby petition for such an extension. If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert & Goetzel PC Deposit Account No. 501505/5500-88700/RCK.

Respectfully submitted,
/Robert C. Kowert/
Robert C. Kowert, Reg. #39,255
Attorney for Applicants

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850
Date: October 18, 2007